If a link metric used in a distance vector routing protocol is jittered 25% due to a bug in the link quality estimation, what is the impact on the route identification and stability. E.g., if the true link latency is 100ms, then the bug will add an error in the range (-25%,25%) of the true value (100ms) to the true link latency resulting in the latency in the range of (75,125) ms. Will the answer be different for link state routing?

Link quality: a way to describe reliability. What percentage of packets will be successfully received by the receiver.

1.

YANG
95% -> actual value (95% * 0.75 to 100%) = (~0.67 to 1)
70%
68%
68%
ALIREZA
75% ->actual value (75% * 0.75 to 75% * 1.25) = (~0.56 to ~0.94)
60%
90%
56%
DUSTIN

YANG -> DUSTIN: (0.67 * 0.56 to 1 * 0.94)
Measurement two 42% <- win
61% <- win
38% <- lose

2.

YANG
50% -> actual value with 25% error = (37% to 62%)
50%
62%
ALI
60% -> actual value with 25% error = (45% to 75%)
60%
75%
DUSTIN

YANG -> DUSTIN: 0.5 * 0.6 = 0.3
30%
46%

Packet header is 20 bytes long. We want to use stop and wait protocol with a window size of one with retry until the packet is acknowledged. The link packet drop rate is 50% in both directions. The latency is 100ms. We need to send 1 MB of data. How long is it likely to take?

Transmission time = 1ms
Packet size = 120 bytes
Payload = 100 bytes
Without error: # of packets to send = 1 MB / 100 bytes = 10000 packets

First:

10000 packets
Total Delay for 1 packet = 1ms + 100ms

Success rate for 1 round trip = 50% * 50% = 25%
Forward ACK

1. Wait 101ms for 10000 packets
2. Sender only receives ACK for only 25% * 10000 packets
3. 75% of packets need to be retransmitted: 7500 packets

Second:

7500 packets

1. Wait 101ms for 7500 packets
2. Sender only receives ACK for only 25% * 7500 packets
3. 75% of packets need to be retransmitted: 5625 packets

Third:

5625 packets

1. Wait 101ms for 5625 packets
2. Sender only receives ACK for only 25% * 5625 packets
3. 75% of packets need to be retransmitted: 5625 * 75% packets

At the end of the third iteration, total time taken will be

101ms * 10000
\+ 101ms * 7500
\+ 101ms * 5625

25% + 25%^2 + 25^3 + = 100%